Generalized Hidden Markov Models for Eukaryotic gene prediction

W.H. Majoros
Recall: Gene Syntax

- **ATG** to **TGA** as the coding segment
- **complete mRNA**
- **exon**
- **intron**
- **start codon**
- **donor site**
- **acceptor site**
- **stop codon**
An HMM is a *stochastic machine* $M = (Q, \alpha, P_t, P_e)$ consisting of the following:

- a finite set of states, $Q = \{q_0, q_1, \ldots, q_m\}$
- a finite alphabet $\alpha = \{s_0, s_1, \ldots, s_n\}$
- a transition distribution $P_t : Q \times Q \mapsto \mathbb{R}$ i.e., $P_t(q_j | q_i)$
- an emission distribution $P_e : Q \times \alpha \mapsto \mathbb{R}$ i.e., $P_e(s_j | q_i)$

**An Example**

\[ M_1 = (\{q_0, q_1, q_2\}, \{Y, R\}, P_t, P_e) \]

\[ P_t = \{(q_0, q_1, 1), (q_1, q_1, 0.8), (q_1, q_2, 0.15), (q_1, q_0, 0.05), (q_2, q_2, 0.7), (q_2, q_1, 0.3)\} \]

\[ P_e = \{(q_1, Y, 1), (q_1, R, 0), (q_2, Y, 0), (q_2, R, 1)\} \]
Generalized HMMs

A GHMM is a stochastic machine $M=(Q, \alpha, P_t, P_e, P_d)$ consisting of the following:

- a finite set of states, $Q=\{q_0, q_1, \ldots, q_m\}$
- a finite alphabet $\alpha=\{s_0, s_1, \ldots, s_n\}$
- a transition distribution $P_t : Q \times Q \mapsto \mathbb{R}$ i.e., $P_t(q_j \mid q_i)$
- an emission distribution $P_e : Q \times \alpha^* \times \mathbb{N} \mapsto \mathbb{R}$ i.e., $P_e(s_j \mid q_i, d_j)$
- a duration distribution $P_d : Q \times \mathbb{N} \mapsto \mathbb{R}$ i.e., $P_d(d_j \mid q_i)$

Key Differences

- each state now emits an entire subsequence rather than just one symbol
- feature lengths are now explicitly modeled, rather than implicitly geometric
- emission probabilities can now be modeled by any arbitrary probabilistic model
- there tend to be far fewer states => simplicity & ease of modification

HMMs & Geometric Feature Lengths

\[ P(x_0 \ldots x_{d-1} \mid \theta) = \left( \prod_{i=0}^{d-1} P_e(x_i \mid \theta) \right) p^{d-1} (1 - p) \]

geometric distribution
Advantages:
* Submodel abstraction
* Architectural simplicity
* State duration modeling

Disadvantages:
* Decoding complexity
Fixed-length states are called *signal states* (diamonds).

Variable-length states are called *content states* (ovals).

Each state has a separate *submodel* or *sensor*.

*Sparse*: in-degree of all states is bounded by some small constant.
Some GHMM Submodel Types

1. WMM (Weight Matrix) \[ \prod_{i=0}^{L-1} P_i(x_i) \]

2. Nth-order Markov Chain (MC) \[ \prod_{i=0}^{n-1} P(x_i | x_0...x_{i-1}) \prod_{i=n}^{L-1} P(x_i | x_{i-n}...x_{i-1}) \]

3. Three-Periodic Markov Chain (3PMC) \[ \prod_{i=0}^{L-1} P(f+i)(\text{mod } 3)(x_i) \]

4. Codon Bias \[ \prod_{i=0}^{n-1} P(x_{\alpha+3i} x_{\alpha+3i+1} x_{\alpha+3i+2}) \]

5. MDD

6. Interpolated Markov Model


Recall: Decoding with an HMM

\[
\phi_{\text{max}} = \arg\max_{\phi} \quad P(\phi \mid S) = \arg\max_{\phi} \frac{P(\phi \land S)}{P(S)} \\
= \arg\max_{\phi} \quad P(\phi \land S) \\
= \arg\max_{\phi} \quad P(S \mid \phi)P(\phi)
\]

\[
P(S \mid \phi) = \prod_{i=0}^{L-1} P_e(x_i \mid y_{i+1})
\]

\[
P(\phi) = \prod_{i=0}^{L} P_t(y_{i+1} \mid y_i)
\]

\[
\phi_{\text{max}} = \arg\max_{\phi} \quad P_t(q_0 \mid y_L) \prod_{i=0}^{L-1} P_e(x_i \mid y_{i+1})P_t(y_{i+1} \mid y_i)
\]
Decoding with a GHMM

\[ \phi_{\text{max}} = \arg\max_{\phi} P(\phi \mid S) = \arg\max_{\phi} \frac{P(\phi \land S)}{P(S)} \]

\[ = \arg\max_{\phi} P(\phi \land S) \]

\[ = \arg\max_{\phi} P(S \mid \phi)P(\phi) \]

\[ P(S \mid \phi) = \prod_{i=1}^{\lvert \phi \rvert - 2} P_e(S_i \mid y_i, d_i) \]

\[ P(\phi) = \prod_{i=0}^{\lvert \phi \rvert - 2} P_t(y_{i+1} \mid y_i)P_d(d_i \mid y_i) \]

\[ \phi_{\text{max}} = \arg\max_{\phi} \prod_{i=0}^{\lvert \phi \rvert - 2} P_e(S_i \mid y_i, d_i)P_t(y_{i+1} \mid y_i)P_d(d_i \mid y_i) \]
Recall: Viterbi Decoding for HMMs

\[
V(i, k) = \begin{cases} 
\max_j V(j, k - 1) P_t(q_i | q_j) P_e(x_k, q_i) & \text{if } k > 0, \\
P_t(q_i | q_0) P_e(x_0 | q_i) & \text{if } k = 0.
\end{cases}
\]

run time: \( \mathcal{O}(L \times |Q|^2) \)
Naive GHMM Decoding

run time: $O(L^3 \times |Q|^2)$
(1) The emission functions $P_e$ for variable-length features are \textit{factorable} in the sense that they can be expressed as a product of terms evaluated at each position in a putative feature—i.e.,

$$\text{factorable}(P_e) \Leftrightarrow \exists f \ P_e(S) = \prod_{i} f(S,i)$$

for $0 \leq i < |S|$ over any putative feature $S$.

(2) The lengths of noncoding features in genomes are geometrically distributed.

(3) The model contains no transitions between variable-length states.
Assumption #3

Variable-length states cannot transition directly to each other.
Each signal state has a *signal sensor*:

![Signal sensor diagram](image)

The “trellis” or “ORF graph”:

![ORF graph diagram](image)
Efficient Decoding via Signal Sensors

sequence: GCTATCGATTCTCTAATCGTCTATCGATCGTG

detect putative signals during left-to-right pass over sequence

signal queues

- ATG’s
- GT’S
- AG’s

insert into type-specific signal queues

sensor n
sensor 2
sensor 1

newly detected signal

elements of the “ATG” queue

trellis links

ATG ATG ATG

GT
Figure 8.17: Operation of the `highestScoringPath()` algorithm for a small portion of a weighted ORF graph. The highest scoring path is shown in bold, and represents the optimal gene parse for this portion of the sequence. LT=left terminus, RT=right terminus.
The Notion of “Eclipsing”

ATG GATGCTACT TGA CGT ACT TAA CTTACCGATCTCT

0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0

in-frame stop codon!
Algorithm 8.1 Eclipsing signals in coding queue $G$ when a stop codon has been encountered at position $p$. $\text{pos}(s)$ is the position of the first base of the signal’s consensus sequence (e.g., the A in ATG). $\text{len}(s)$ is the length of the signal’s consensus sequence (e.g., 3 for ATG).

procedure eclipse(ref $G, p$
1. foreach $s \in G$ do
2. $\omega \leftarrow (\text{pos}(s) + \text{len}(s) - p) \mod 3$;
3. $\text{eclipsed}_s[\omega] \leftarrow \text{true}$;
4. if $\text{eclipsed}_s[(\omega + 1) \mod 3]$ and
5. $\text{eclipsed}_s[(\omega + 2) \mod 3]$
6. then drop($s, G$);
Bounding the Number of Coding Predecessors

**Figure 8.6** Number of potential coding predecessors ($\Gamma_{bc}$) as a function of sequence length (in megabases). Data are from a 3 Mb human DNA sequence. Linear regression resulted in a slope of $-0.000001$, demonstrating that the number of coding predecessors does not increase without bound; the intercept indicated 27.3 predecessors on average.
Geometric Noncoding Lengths (Assumption #2)

Figure 8.5 Intron length distribution for Homo sapiens (solid line) and a geometric distribution (dashed line) with parameter $p=0.002$. 
Prefix Sum Arrays for Emission Probabilities

(Assumption #1: factorable content sensors)

\[ \alpha_\gamma : \]

\[ \text{start codon sensor window} \]

\[ \text{putative exon } E \]

\[ \text{donor site sensor window} \]

\[ ACGTTACGGCAATGAAATCGCAAGCGCTATATATGTGCTGCTAGCGTATCGTA \]

\[ \alpha_\gamma [b] \]

\[ \alpha_\gamma [e] \]

\[ \log P_e(S_{b+1:e} \mid \text{exon, } \omega) = \alpha_\gamma [e] - \alpha_\gamma [b] \]

**Figure 8.9** Using a prefix sum array to compute the emission probability for a putative exon. Subtraction is performed between array elements at either end of the \([b,e]\) interval, which covers only the portion of the exon not covered by either signal sensor.
Bounding the Number of Noncoding Predecessors

**Theorem 8.1 (Burge's noncoding predecessor theorem)**
Suppose that for some signal $s_j \in \Gamma_{en}$, the optimal noncoding predecessor was found to be some $s^* \in \Gamma_{bn}$. If the next signal encountered after $s_j$ is some $s_i \in \Gamma_{en}$, then under assumptions (1) and (2) above, $s^*$ is also the optimal predecessor (of its type and phase) for $s_i$.

**Figure 8.7** Once an optimal noncoding predecessor $s^*$ is identified for signal $s_j$, no signal $z$ preceding $s^*$ can be selected as a predecessor of a later signal $s_i$, assuming a geometric length distribution and a factorable content scoring function.

Algorithm 8.3 Overview of the PSA decoding algorithm. See text for details.

**procedure** PSA($S, \theta$)
1. Initialize arrays via Equations 8.9 & 8.10
2. At each position along the sequence do:
3. Perform eclipsing via Algorithm 8.1
4. Apply signal sensors at current location
5. If a putative signal $s_i$ is detected then:
6. Link $s_i$ back to optimal predecessors via Eq. 8.14
7. Append $s_i$ to appropriate signal queues
8. Form the optimal parse $\phi^*$ via Algorithm 8.2
9. Convert $\phi^*$ to a set of gene predictions

run time: $\mathcal{O}(L|Q|)$ (assuming a sparse GHMM topology)
Algorithm 8.2 Reconstruction of the optimal parse by tracing back through trellis links. Parameters are the selected right-terminus signal $s$ and its chosen phase $\omega$. Returns a stack of signals constituting the optimal parse, with the top signal at the beginning of the parse and the bottom signal at the end. $\text{exon_length}(p, s)$ denotes the number of coding nucleotides between signals $p$ and $s$.

procedure traceback$(s, \omega)$
1. stack $K$;
2. push $K, s$;
3. while $\neg$left_terminus$(s)$ do
4. \hspace{1em} $p \leftarrow \text{pred}(s, \omega)$;
5. \hspace{1em} push $K, p$;
6. \hspace{2em} if type$(p) \in \{\text{ATG, TAG}\}$ then $\omega \leftarrow 0$;
7. \hspace{2em} elseif type$(p) = \text{AG}$ then
8. \hspace{3em} $\omega \leftarrow (\omega - \text{exon_length}(p, s)) \mod 3$;
9. \hspace{1em} $s \leftarrow p$;
10. return $K$;
DSP = **Dynamic Score Propagation**: we incrementally propagate the scores of all possible partial parses up to the current point in the sequence, during a single left-to-right pass over the sequence.

DSP Decoding

Algorithm 8.4 Overview of the DSP decoding algorithm. See text for details.

procedure DSP$(S, \theta)$
1. At each position along the sequence do:
2. Evaluate all content sensors
3. Update accumulators with content scores
4. Perform eclipsing via Algorithm 8.1
5. Allow mature signals to graduate from their holding queues
6. Apply signal sensors at current location
7. If a putative signal $s_i$ is detected then:
8. Link $s_i$ back to optimal predecessors
9. Propagate appropriate queue elements
10. Append $s_i$ to appropriate holding queues
11. Form the optimal parse $\phi^*$ via Algorithm 8.2
12. Convert $\phi^*$ to a set of gene predictions

run time: $\mathcal{O}(L \times |Q|)$
(assuming a sparse GHMM topology)
PSA vs. DSP Decoding

Theorem 8.2 (Equivalence of PSA and DSP) Let S be a sequence and θ a set of model parameters for a GHMM. Then given parses \( \phi^*_{\text{PSA}} = \text{PSA}(S, \theta) \) and \( \phi^*_{\text{DSP}} = \text{DSP}(S, \theta) \) selected under the PSA and DSP decoding algorithms, respectively, we have \( \phi^*_{\text{PSA}} = \phi^*_{\text{DSP}} \).

**Space complexity:**

PSA: \( \mathcal{O}(L \times |Q|) \)

DSP: \( \mathcal{O}(L + |Q|) \)

<table>
<thead>
<tr>
<th></th>
<th>RAM/state (Mb)</th>
<th>seconds/state</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>0.95</td>
<td>2.8</td>
</tr>
<tr>
<td>PSA</td>
<td>14</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Table 8.2 Comparison of memory and time requirements of the PSA vs. DSP decoding algorithms on a sample 922 kb sequence. DSP requires far less memory, while achieving the same speed as PSA. Adapted from Majoros et al., 2005a.
Modeling Isochores

I: \{G,C\} density \in [0\%, 43\%],
II: \{G,C\} density \in (43\%, 51\%],
III: \{G,C\} density \in (51\%, 57\%],
IV: \{G,C\} density \in (57\%, 100\%].

Explicit Modeling of Noncoding Lengths

Figure 8.14: Explicit length modeling for noncoding features. The observed distribution of Arabidopsis intron lengths (solid line) is shown with a geometric distribution (dashed line). The geometric distribution is a reasonably good fit for lengths > 100 bp. Below 100 bp we can use an explicit length distribution, as long as the intron queue stores all donor sites fewer than 100 bp away from the current position.

Figure 8.15: Explicit length modeling for introns. The short intron model is capable of generating introns of length up to some maximum, $L_{\text{short}}$, whereas the long intron model can generate only introns longer than this.

MLE Training for GHMMs

\[
\theta_{MLE} = \arg \max_{\theta} \left( \prod_{(S,\phi) \in T} P(S,\phi) \right)
\]

\[
= \arg \max_{\theta} \left( \prod_{(S,\phi) \in T} \prod_{y_i \in \phi} P_e(S_i \mid y_i, d_i) P_t(y_i \mid y_{i-1}) P_d(d_i \mid y_i) \right)
\]

\[
= \arg \max_{\theta} \left( \prod_{(S,\phi) \in T} \prod_{y_i \in \phi} P_t(y_i \mid y_{i-1}) P_d(d_i \mid y_i) \prod_{j=0}^{\lvert S_i \rvert-1} P_e(x_j \mid y_i) \right)
\]

estimate via labeled training data, as in HMM

\[
a_{i,j} = \frac{A_{i,j}}{\sum_{h=0}^{\lvert Q \rvert-1} A_{i,h}}
\]

construct a histogram of observed feature lengths

estimate via labeled training data, as in HMM

\[
e_{i,k} = \frac{E_{i,k}}{\sum_{h=0}^{\lvert \alpha \rvert-1} E_{i,h}}
\]
Maximum Likelihood vs. Conditional Maximum Likelihood

\[ \theta_{\text{MLE}} = \arg\max_{\theta} \left( \prod_{(S, \phi) \in T} P(S, \phi) \right) \]

\[ = \arg\max_{\theta} \left( \prod_{(S, \phi) \in T} \prod_{y_i \in \phi} P_e(S_i | y_i, d_i) P_t(y_i | y_{i-1}) P_d(d_i | y_i) \right) \]

maximizing \( P_e, P_t, P_d \) separately does not maximize the entire expression

\[ \theta_{\text{CML}} = \arg\max_{\theta} \left( \prod_{(S, \phi) \in T} P(\phi | S) \right) \]

\[ = \arg\max_{\theta} \left( \prod_{(S, \phi) \in T} \prod_{y_i \in \phi} P_e(S_i | y_i, d_i) P_t(y_i | y_{i-1}) P_d(d_i | y_i) \right) \frac{P(S)}{\prod_{y_i \in \phi} P(S)} \]

maximizing \( P_e, P_t, P_d \) separately does not maximize the entire expression

\[ \therefore \text{MLE} \neq \text{CML} \]
Discriminative Training of GHMMs

\[ \theta_{\text{discrim}} = \arg\max_{\theta} (\text{accuracy on training set}) \]
Discriminative Training of GHMMs

- mean intron, intergenic, and UTR lengths
- transition probabilities
- exon “optimism” (section 7.3.4)
- sizes of all signal sensor windows
- locations of consensus regions within signal sensor windows
- emission orders for Markov chains and other models
- sensitivity of signal thresholds (when thresholding is used—section 8.3.1)
- number of signal boosting iterations to utilize during signal training (section 11.1)
- skew and kurtosis of exon length distributions (section 2.6)

<table>
<thead>
<tr>
<th></th>
<th>nucleotide</th>
<th>exon</th>
<th>gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>gradient ascent</td>
<td>94%</td>
<td>81%</td>
<td>48%</td>
</tr>
<tr>
<td>MLE</td>
<td>90%</td>
<td>71%</td>
<td>33%</td>
</tr>
</tbody>
</table>

Table 8.3: Gradient ascent vs. maximum likelihood estimation for GHMM training. Both protocols were applied to 1000 training and 1000 (distinct) test genes from Arabidopsis thaliana. Metrics are, left to right: nucleotide SMC, exon F-measure, and gene sensitivity. Results from (Majoros and Salzberg, 2004).
• GHMMs generalize HMMs by allowing each state to emit a subsequence rather than just a single symbol

• Whereas HMMs model all feature lengths using a geometric distribution, coding features can be modeled using an arbitrary length distribution in a GHMM

• Emission models within a GHMM can be any arbitrary probabilistic model ("submodel abstraction"), such as a neural network or decision tree

• GHMMs tend to have many fewer states => simplicity & modularity

• When used for parsing sequences, HMMs and GHMMs should be trained discriminatively, rather than via MLE, to achieve optimal predictive accuracy